

Docket No. 3271.2.1
The Code Corporation

UNITED STATES PATENT APPLICATION
FOR

**EXTENSIBLE APPLICATION INTERFACE USING MACHINE-
READABLE GRAPHICAL CODES**

Inventor(s): **Paul J. Hepworth**, a citizen of the United States, residing
in Riverton, UT

Dimitri V. Yatsenko, a citizen of Ukraine, residing in
Sandy, UT

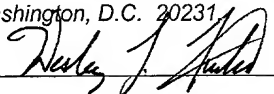
Darren P. Smith, a citizen of the United States, residing in
Orem, UT

Assignee: The Code Corporation
11814 S. Election Dr., Suite 200
Draper, UT 84020

"Express Mail" Label Number EV001614345US

Date of Deposit Nov. 20, 2001

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents, Box No Fee Patent Application,, Washington, D.C. 20231.

 Nov. 20, 2001
Date

EXTENSIBLE APPLICATION INTERFACE USING MACHINE-READABLE GRAPHICAL CODES

Inventors:

5

Paul J. Hepworth, Dimitri V. Yatsenko and Darren P. Smith

BACKGROUND OF THE INVENTION

RELATED APPLICATIONS

10

This application is related to and claims priority from U.S. Patent Application Serial No. 60/279,356 filed March 27, 2001, for "Extensible Application Interface Using Machine-Readable Graphical Codes," with inventors Paul Hepworth, Dimitri Yatsenko and Darren Smith, which is incorporated herein by reference.

FIELD OF THE INVENTION

15

The present invention relates generally to the field of graphical-code reading computer systems. More specifically, the present invention relates to a system and method for providing information from machine-readable graphical codes to applications on a computer system.

DESCRIPTION OF RELATED BACKGROUND ART

20

Computer technology has entered many areas to simplify manual tasks and to make information more readily available. Most people use several computer programs every day that greatly simplify their work day. In addition, through the use of a computer, vast amounts of information are readily available. Computer software and electronic information sources are typically found on storage media or storage

devices such as hard drives, CD-ROMs, DVD-ROMs, etc., on a local computer, on a local computer network or a global computer network, such as the Internet.

Computer programs can be used for many purposes including assisting a person in performing his or her job. For example, word processors help computer users prepare documents, spreadsheet programs help users perform accounting functions and numerical analysis, diagnostic programs assist users in diagnosing problems, etc. There are many programs available to help users with almost any need they may have. Generally, computer programs need some type of manual input to help a user, from simply starting the program to entering a significant amount of input.

Before a user can access relevant electronic information, he or she usually needs to enter some input before helpful information becomes available. By way of example, many computer users, when looking for particular information, will use the World Wide Web (the "Web") to find information. Typically users will begin their search for information by using a search engine on the Web. To perform a search, a user first enters one or more search terms. Typically, a user will then browse the results by clicking on various links and reading through the information found. After some manual browsing, the user often finds the relevant information. Finding and accessing electronic information from a CD-ROM or from a hard drive is similar in that some manual searching and browsing of data is required.

Usually computer programs and/or electronic information relates to a particular product, item or task. As illustrated, the computer user often needs to provide input to use the program or to access the information. For example, the

particular product, item or task may require a user to start a particular computer program, access certain electronic information, enter particular input using the keyboard or mouse, complete an online form, etc. It would be beneficial if means were provided to enable more of the manual data entry and/or manual application
5 launching to be achieved automatically.

BRIEF DESCRIPTION OF THE DRAWINGS

Non-exhaustive embodiments of the invention are described with reference to the figures, in which:

10 FIG. 1 is a block diagram of an embodiment of a system for providing data from a graphical code reading device to a software client module on a computer system;

FIG. 2 is a block diagram of another embodiment of a system for providing data from a graphical code reading device to a software client module on a
15 computer system;

FIG. 3 is a block diagram of an embodiment of a system for providing data from a graphical code reading device to a web page;

FIG. 4 is a block diagram of an embodiment of graphical code data;

FIG. 5 is a block diagram of another embodiment of graphical code
20 data;

FIG. 6 is a block diagram of a further embodiment of graphical code data;

FIG. 7 is a block diagram of another embodiment of graphical code data;

FIG. 8 is a block diagram of an embodiment of an association list;

FIG. 9 is a block diagram of hardware components that may be used in
5 an embodiment of a computer;

FIG. 10 is a flow diagram of an embodiment of a method for providing graphical code data to a requesting software component;

FIG. 11 is a flow diagram of another embodiment of a method for providing graphical code data to a requesting software component;

FIG. 12 is a flow diagram of an embodiment for communicating the
10 graphical code data from a server module to a client module; and

FIG. 13 is a flow diagram of an embodiment for communicating graphical code data to assist in processing a web-related document.

15

DETAILED DESCRIPTION

A system is disclosed that includes a graphical code reading device that scans a graphical code and provides graphical code data. A computer is in electronic communication with the graphical code reading device and receives the graphical code data from the graphical code reading device. (The graphical code
20 reading device may decode the data from the graphical code image and transmit the decoded data to the computer; or the reading device may transmit the image, and the computer may decode the data from the image.) The computer may include a processor, a memory in electronic communication with the processor and a

communications port in electronic communication with the processor for communicating with the graphical code reading device. An association may include a data type and a software module identifier. A driver may be used to receive the graphical code data and obtain the software module identifier through use of the data type to provide the graphical code data to the software module identified by the software module identifier.

In one embodiment the association is included in an association list. The association list may be created statically or dynamically. The association list may include a number of data types and software module identifiers.

The graphical code data may have different formats depending on the particular embodiment. For example, the graphical code data may include a data field or multiple data fields.

Embodiments of the driver may also include a server module. The server module operates to receive the graphical code data and determine a data type of the graphical code data. Using the data type the server module may obtain the software module identifier from the association. If the software module identified by the software module identifier is not running, the software module may be launched. The graphical code data may be provided to a client module of the software module.

Also disclosed is a method for providing data from a graphical code reading device to a software module. Graphical code data is received from a graphical code reading device in electronic communication with a computer and the data type of the graphical code data is determined. A software module identifier is

obtained through use of the data type and through use of an association list. If the software module identified by the software module identifier is not running, the software module may be launched. The graphical code data is then provided to the software module. A computer-readable medium is disclosed for storing program
5 data where the program data comprises executable instructions for implementing the method.

In another embodiment, a system is disclosed for providing data from a graphical code reading device to a web page. An embodiment of the system includes a graphical code reading device that scans a graphical code and provides
10 graphical code data. A computer is in electronic communication with the graphical code reading device and receives the graphical code data from the graphical code reading device. The computer may include a processor, a memory in electronic communication with the processor and a communications port in electronic communication with the processor for communicating with the graphical code
15 reading device. An association may include a data type and a software module identifier. A driver may be used to receive the graphical code data and obtain the software module identifier through use of the data type. The driver may further provide the graphical code data to a browser extension module identified by the software module identifier. The browser extension module provides the graphical
20 code data to the web page.

In an embodiment herein, the web page may include a form having form fields. Further, the graphical code data may include multiple data fields that correspond to the form fields such that the form is automatically completed.

In another embodiment, the driver provides the graphical code data to the browser extension module by having the driver send the graphical code data to a client module embedded in the browser extension module. Then the client module calls a browser extension callback function to pass the graphical code data to the browser extension module.

Also disclosed herein is a serialization process that may be used to avoid losing transactions. The serialization process may cause the driver to wait for a period of time. One method for timing the period of time is by causing the driver to wait until a message is sent from the client module to stop the wait.

Referring now to FIG. 1, there is shown an embodiment of a system 100 for providing data from a graphical code reading device 102 to a software client module 104 on a computer system 106. The computer 106 is in electronic communication with the graphical code reading device 102. The graphical code reading device 102 scans a graphical code 108 to obtain or provide graphical code data to the computer 106. Various types of graphical codes 108 may be used with systems and methods herein. For example, bar codes or matrix codes may be used as graphical codes 108. Of course, any other graphical code 108 that may be scanned may be used with embodiments herein.

The graphical code reading device 102 may be connected to or integrated with the computer 106. If graphical code reading device 102 is connected to the computer 106, the connection may be wireless or wired, or may be continuous or intermittent.

The computer 106 shown in FIG. 1 may be a personal computer. Personal computers are commercially available and known by those skilled in the art. Components typically found in a computer 106 will be discussed below in relation to FIG. 9.

5 In operation, a server module 110 receives the graphical code data from the graphical code reading device 102 and sends the graphical code data to any client modules 104 that requested the data. The server module 110 may identify any requesting client modules 104 through use of an association list 112. The association list 112 will be more fully discussed below. Generally, the
10 association list 112 includes a list of data types that may be received in the graphical code data. The association list 112 also includes a list of client modules 104 that have requested each data type so that when the server module 110 receives the graphical code data, it 110 may identify what client modules 104 have requested that data type by using the association list 112.

15 It will be appreciated by those skilled in the art that the inventive principles herein may be applied and implemented in a wide variety of ways. Different examples of such implementations will be shown through embodiments herein to illustrate the versatility of the present invention and that it may be used in many different embodiments all within the scope of the present invention.

20 FIG. 2 illustrates an embodiment where the server module 210 is included as part of driver software 214. The association list 112 may be embodied in various forms and in various ways. For example, as shown in FIG. 2, the driver software 214 may include an integrated association list 216 as well as the server

module 210. In addition, the association list 212 may also be included in the embodiment 200. The association lists 212, 216 may be static or dynamic. The association lists 212, 216 are expandable and may be modified. New data types may be created and added to the association lists 212, 216 at any time making the system extensible. The lists 212, 216 may be based on tables and/or may be created or updated when applications are installed and/or when they are executed.

In the embodiment 200 shown in FIG. 2, the driver software 214 receives the graphical code data from the graphical code reading device 102 and sends the graphical code data to any client modules 204 that requested the data. The server module 210, as part of the driver software 214, identifies any requesting entities through use of the integrated association list 216 and/or the association list 212.

The client module 204 may be used in combination with, be part of, be linked to, or otherwise operate in conjunction with application software 218. The client module 204 may be embedded in the application 218 (e.g., statically linked to the application 218 or dynamically linked to the application).

The server module 210 may send the graphical code data to the client module 204. Alternatively, the server module 210 of the driver software 214 may send the graphical code data to the application software 218 that forwards the graphical code data to the client module 204. Those skilled in the art will appreciate that the graphical code data may be communicated to the client module 204 directly or indirectly. In the embodiment shown in FIG. 2, the client module 204 may

communicate with the driver software 214 using standard interprocess communication techniques such as TCP sockets.

When a graphical code 108 is read, the driver software 214 receives the graphical code data and determines the appropriate application 218 to send the data to. The driver software 214 may launch the application 218 if it isn't already running. The driver software 214 may send the graphical code data to the client module 204 using interprocess communication techniques. The client module 204 receives the data over the socket and may call an application-defined function (not shown) that takes appropriate application-specific action.

Referring to FIG. 3, embodiments herein may also be used in combination with a web browser 322 and web pages 324. As shown, the client module 304 of the embodiment in FIG. 3 may be used in combination with a browser extension module 320. Browser extensions 320 are known by those skilled in the art.

The browser extension module 320 is used with the web browser to extend its capabilities or functionality. Typically the web browser 322 is used to display web pages 324. Browser extension modules 320 may add functionality to enable or to allow the display or processing of certain web pages. Examples of browser extension modules 320 are Netscape plugins, ActiveX controls, etc.

In the embodiment of FIG. 3, the client module 304 may be used to send graphical code data to web browser 322 and/or to a web page 324. For example, some web pages display forms and provide processing for the form. The client module 304 may receive data to fill out forms from web pages 324. For

example, a manufacturer may provide support for a product through its web site. To obtain support, the manufacturer's web site may include a web page 324 with a form to complete. Through the use of the embodiment shown in FIG. 3, the user may simply scan a graphical code 108 to complete the form, rather than have to manually complete the form. The code 108 scanned may be placed on the product (not shown).

The graphical code reading device 102 provides graphical code data to the computer 306, and more particularly to the driver software 214. The server module 210 may receive the graphical code data and send it to the appropriate requesting software.

In order to receive the relevant graphical code data, the client module 304 may have notified the server module 210 of its request for any graphical code data relating to the product. As a result, when the graphical code data is provided to the server module 210, the server module 210 uses the integrated association list 216 and/or the association list 212 to ascertain where the data should be sent. From the lists 216, 212, the server module 210 determines that the graphical code data should be sent to the client module 304 used in combination with the browser extension module 320.

The client module 304 receives the graphical code data and processes the data and operates to complete the form provided by the web page 324 and web browser 322. To complete the form, the client module 304 may pass all of the graphical code data to the web page 324 at once, or it may pass portions of the graphical code data to the web page 324, as required. For example, if the form

contained different fields, the client module 304 may provide the field data (from the graphical code data) to the web page 324 a field at a time. Of course, it will be appreciated by those skilled in the art that web pages 324 may include a variety of components that can be used to present or process a form including, but not limited to, scripts, Java, ActiveX, extensions, plugins, etc. Thus, the client module 304 may pass the fields to many different types of program code to complete the form processing.

In one embodiment, the web page 324 may include tags that activate the browser extension module 320. The browser extension module 320, or more specifically the client module 304 of the browser extension module 320, may communicate with the server module 210. Once the browser extension module 320 receives the graphical code data, it 320 may communicate the data to the web page 324 by calling script functions (not shown) contained within the web page. Thus, the embodiment illustrated in FIG. 3 may be used to fill out web forms, to select data, to retrieve data, to assist or facilitate electronic commerce transactions, etc.

The embodiment of Figure 3 may easily be used with other extensible applications besides web browsers 322 and browser extension modules 320. The embodiment of Figure 3 may more generally use an extensible application (not shown) along with application extension modules (not shown) in the same manner illustrated by the web browser 322 and browser extension module 320. For example, word processors, spreadsheets, multimedia presentation packages, etc., may be used as extensible applications. Those skilled in the art will appreciate that

many other extensible applications may be used with the embodiments disclosed herein.

Referring now to FIG. 4, the graphical code data 402 may have different formats. The graphical code data 402 may include a data type 404
5 indicating what type of data 406 is included in the graphical code data 402. FIG. 4 illustrates graphical code data 402 including one piece of data 406 and one data type 404 encoded along with the data 406.

Data type identifiers 404 may be text, numbers, etc. Some examples of possible data type identifiers 404 are as follows: first_name, last_name, addr1,
10 addr2, ssn, URL, program, email, file, credit_card_expir, submit, clear, print, update, credit_card_num, 12, C43FE89A, 1020, 1021 and credit_card_type.

Data type identifiers 404 do not need to be defined in advance of when the driver software 214 or graphical code reading device 102 are produced. An application 218 or client module 204 may use many different identifiers 404. It is
15 only necessary that the creators of the graphical codes 108 and the applications 218 and client modules 204 agree on the meaning of the type identifier.

FIG. 5 depicts graphical code data 502 that includes a plurality of data types and data. A first type 504 may indicate what type of data is included in the first data 506. A second type 508 may indicate what type of data is included in the
20 second data 510, and a third type 512 may indicated what type of data is included in the third data 514, etc. The graphical code data 502 illustrated in FIG. 5 is an example of what graphical code data used in combination with a browser extension client module 304, as shown in FIG. 3, may include. The first type 504 and first data

506 may correspond to one field of a form of a web page 324. The second type 508 and second data 510 may correspond to a second field of the form of a web page 324, and so forth.

The graphical code data 502 of FIG. 5 illustrates that the different fields
5 may be dispatched individually, each field going to a different application. Alternatively, and as discussed above, the fields may be dispatched together and be sent to the same application.

Of course, it will be appreciated by those skilled in the art that the graphical code data 602 may not have an embedded data type. As shown in FIG. 6,
10 the graphical code data 602 may simply include the data 606. If this embodiment of graphical code data is used, the server module 110 may respond in a number of ways including, but not limited to, examining the data 606 to determine what type of data it is, assuming that the data 606 is of a particular data type, passing the data 606 on to another component, determining the data type according to the symbology
15 of the graphical code, etc. In addition, the server module 110 may assume that the data 606 is of a single generic type or that it is of a type derived from the encoding format of the graphical code 108 (e.g., UPC, "code 39 text", etc.).

The server module 110 may examine the data 606 to determine what type of data it is. The server module 110 or driver 214 may use specific substrings,
20 regular expressions or patterns in the data to infer a data type. For example, "www" or "http" may indicate a URL type, ".exe" may indicate a program type, ".txt" or ".doc" may indicate a document type, nnn-nn-nnnn may indicate a social security number

type, etc. Those skilled in the art will appreciate the ways in which data may be examined to determine the data type.

FIG. 7 illustrates graphical code data 702 that includes both fields with data types and fields without data types. A data type 704 may be included to indicate what type of data 706 is included. Additional data 708 may also be included as part of the graphical code data 702. As shown, the additional data 708 may be included without an associated data type field. The server module 110 may be configured to handle the graphical code data 702 in a variety of ways. The server module 110 may assume that the additional data 708 is of the same type as indicated by data type 704. Alternatively, the server module 110 may assume that if an additional data 708 field is present, it is of a particular data type but not the same as the data type 704. Furthermore, the server module 110 may examine the additional data 708 to determine its type, may simply pass the additional data 708 to a particular client module 304, may save the additional data 708 until a later time when it may be correctly routed, etc. As shown, there are a number of ways in which the graphical code data may be formatted and processed.

FIG. 8 illustrates an embodiment of an association list 812. The association list 112, 212 and the integrated association list 216 may have the same format or may have different formats. In function, the association list 812 serves to indicate what data types should be sent to which components or applications. Those skilled in the art will appreciate how this may be implemented in different ways. The association list 812 illustrated herein includes a plurality of data types 804. With each data type 804 is listed software module identifier(s) identifying

software components that have requested any graphical code data of that data type 804 be sent to it. As shown, application A1 814a, application A2 816a and application A3 818a have all requested that any graphical code data of type data type A 804a be sent to them. The application names 814, 816, 818, etc., may correspond to the client modules 104, 204, 304 or may correspond to an associated application (such as a browser extension module 320 or a web browser 322). Data type B 804b has been requested by application B1 814b. Data type C 804c has been requested by application C1 814c. Further, as shown, data type D 804d has been requested by application D1 814d and application B1 816d. As shown by application B1 814b, 816d, one application may request more than one data type. Finally, data types E and F 804e, 804f have been requested by applications E1 and F1 814e, 814f, respectively.

FIG. 9 is a block diagram of hardware components that may be used in an embodiment of a computer 906 used in combination with the graphical code reading device 102. The computer 906 is used in combination with the graphical code reading device 102 to read in the graphical codes 108. The embodiment of the computer 906 shown in FIG. 9 communicates with the graphical code reading device 102 through the reading device interface 930. The reading device interface 930 may be a standard communications port typically found on a computer 906, or it may be a specialized interface card provided along with the graphical code reading device 102.

Many different types of computer systems may be used to implement the computer 906 illustrated herein. The diagram of FIG. 9 illustrates typical

components of a computer 906 including a processor 932, memory 934, a storage device 936, an input device 938, and an output device 940.

One or more communication ports 942 may also be included in the computer 906. It will be appreciated by those skilled in the art that more components may be included in the computer 906. For example, several input devices 938 may be included, such as a keyboard, a mouse, a joystick, a touch screen, etc. In addition, several output devices 940 may be included such as a monitor, speakers, a printer, etc. Thus, those skilled in the art will appreciate that additional components may be added to the computer 906 without detracting from the functionality to serve as a computer 906.

The computer 906 may be a conventional desktop computer. Desktop computers are commercially available. However, it will be appreciated by those skilled in the art that the computer 906 is a broadly defined digital computer. A computer 906, as used herein, is any device that includes a digital processor capable of receiving and processing data. A computer 906 includes the broad range of digital computers including microcontrollers, hand-held computers, personal computers, servers, mainframes, supercomputers, and any variation or related device thereof. In current design, the computer 906 is typically an IBM-compatible personal computer running the Linux or Microsoft Windows 95/98/2000 or NT operating system. Of course, other types of computers with different operating systems may be used. For example, an Apple computer or a UNIX workstation may be used as the computer 906.

FIG. 10 illustrates an embodiment of a method for providing graphical code data to a requesting software component. The graphical code reading device 102 reads 1002 graphical code data and then provides 1004 the graphical code data to the computer 106. The server module receives 1006 the graphical code data. 5 Once the server module 210 has the graphical code data, it may determine 1008 the data type of the graphical code data. The server module accesses 1010 an association list 212, 216 to determine whether any software modules have requested the data type. The server module obtains 1012 software module identifiers from the association lists 212, 216 using the data type. Finally, the server 10 module sends 1014 the graphical code data to the software modules identified by the software module identifiers.

FIG. 11 illustrates another embodiment of a method for providing graphical code data to a requesting software component and illustrates that various changes can be made by those skilled in the art. The graphical code reading device 15 102 reads 1102 graphical code data. In the embodiment of FIG. 11, the graphical code reading device then decodes 1104 the raw data from the graphical code. In addition, the graphical code reading device may store 1106 the graphical code data for later communication to the computer 106. For example, an embodiment may be configured to store graphical code data for a certain time period, or until a certain 20 amount of graphical code data is reached, before sending the graphical code data to the computer 106.

The graphical code reading device provides 1108 the graphical code data to the computer 106. The server module receives 1110 the graphical code

data. If the graphical code reading device 102 had not decoded 1104 the raw data, the driver 214 or server module 210 may decode the raw data. Once the server module 210 has the graphical code data, it may determine 1112 the data type of the graphical code data.

5 As illustrated herein, there may be different kinds of association lists. For example, there may be an association list 212 and/or an integrated association list 216. The association list 212 may be a dynamic association list (based on data types desired by currently running applications), and the integrated association list 216 may be a static association list (based on default applications). In an
10 embodiment using both integrated and dynamic association lists 216, 212, the server module accesses 1114 the dynamic association list to determine whether any software modules have requested the data type. In addition, the server module may access 1116 the static association list to determine whether any other software modules are to receive the data type. The server module obtains 1118 software
15 module identifiers from the dynamic and static association lists using the data type. Finally, the server module sends 1120 the graphical code data to the software modules identified by the software module identifiers.

FIG. 12 illustrates a flow diagram of the communicating of the graphical code data from the server module 210 to the client module(s). If the
20 software module identified by the software module identifier is not running, the server module 210 may launch 1202 the software module. The client module 204 of the software module started connects 1204 to the server module 210 for electronic communication. The server module 210 sends 1206 the graphical code data to the

client module 204. The client module 204 receives 1208 the graphical code data and provides the graphical code data to the appropriate software module. One way in which the client module 204 may provide the graphical code data to the appropriate software module is by calling an application-defined callback function
5 that takes appropriate application-specific action. The software module then processes 1210 the graphical code data.

FIG. 13 is a flow diagram illustrating the use of an embodiment to communicate graphical code data to assist in processing a web-related document. Graphical code data may be sent to a web-related document to provide various
10 pieces of data or information including, but not limited to, product numbers, names, addresses, telephone numbers, credit card numbers, billing information, shipping information, submission code, confirmation codes, etc.

The method illustrated in FIG. 13 may be used in combination with the embodiment shown in FIG. 3. A web page 324 is provided 1302 to a web browser
15 322 for display. The web page 324 may include a component or instructions that cause a browser extension 320 to be loaded. For example, the web page 324 may include a tag that activates the browser extension 320. The client module 304 is started 1304 by the loading of the browser extension 320.

The client module 304 is provided 1306 with data types to be
20 associated with the web page 324 so that the client module 304 may request these data types. The client module 304 may be configured to call a function or functions provided by the web page 324 to return the data types needed. Alternatively, the client module 304 may be configured to examine the web page 324 and determine

the types of data to request. Other means may be used to determine what data types to request. For example, the client module 304 may post an event which a script responds to and calls a function of the browser extension 320 to set the data types. The client module 304 communicates 1308 the data types to the server
5 module 210 to dynamically associate the data types with the browser extension 320.

After a user has scanned a graphical code 108 with the graphical code reading device 102, the server module 210 receives 1310 the graphical code data intended for use with the web page 324. The server module 210 determines 1312 the data type of the graphical code data. The server module 210 may then access
10 1314 the dynamic association list to identify the dynamically associated browser extension module 320. Using the association list, the server module 210 is informed that the browser extension module 320 has requested the data type. The server module 210 then sends 1316 the graphical code data to the browser extension client module 304.

15 The client module 304 then provides 1318 the graphical code data to the browser extension for processing. One way in which the client module 304 may provide the data to the extension 320 is by calling the browser extension's callback function. The browser extension module 320 provides 1320 the graphical code data to the web page 324 and the web page processes the data. Those skilled in the art
20 will appreciate the various ways in which the graphical code data may be provided to the web page 324. For example, a callback function may pass the data to the web page by (a) calling the web page's corresponding callback script function and passing the data as parameters, (b) posting an event to which the web page script

responds by calling a browser extension function to retrieve the data, or (c) posting an event which includes the data as part of the event. Of course, data may be passed to the web page by various other means, as will be appreciated by those skilled in the art. Once the web page 324 receives the data, it takes the appropriate

5 web-page specific action according to the data received, such as filling in fields on a form, submitting a form, loading a frame, refreshing a page, controlling multimedia content, etc.

Those skilled in the art will appreciate that a similar technique to that of using an extension module with a web-page script can be applied to other types of application software; for example, a Microsoft Word document, through a Visual

10 Basic for Applications (VBA) script, can load an extension module containing an embedded client that can receive data read from graphical codes and pass this data to the VBA script for processing in a manner appropriate to the particular document. As another example, the same technique could be applied to a Microsoft PowerPoint

15 presentation to enable it to respond to graphical codes, which could cause a particular slide to be displayed when the graphical code reader reads a particular code.

Sometimes problems may occur when scanning codes for a web page if the user reads another code before the web page has finished loading and the

20 transaction may be lost. A process of serialization may be used to prevent this type of problem. The serialization process may include the following steps. The graphical code containing a serialized URL is read and decoded and passed to the driver software 214. The driver software determines that the data is a URL and

While specific embodiments and applications of the present invention
15 have been illustrated and described, it is to be understood that the invention is not
limited to the precise configuration and components disclosed herein. Various
modifications, changes, and variations which will be apparent to those skilled in the
art may be made in the arrangement, operation, and details of the methods and
systems of the present invention disclosed herein without departing from the spirit
20 and scope of the invention.

What is claimed is: